

## 1、New操作符具体干了什么？

- a) 创建实例对象，this变量引用该对象，同时还继承了构造函数的原型；
- b) 属性和方法被加入到this引用的对象中；
- c) 新创建的对象由this所引用，并且最后隐式的返回 this。

## 2、下面程序输出结果是？

```
1 for (i=0, j=0; j<6, i<10; i++, j++) {  
2     k = i + j;//9 + 9  
3 }  
4 console.log(k);//18  
5 //，运算符返回值是最后那个表达式的值；
```

## 3、如何检查一个变量是String类型

- typeof(str) === 'string'
- Object.prototype.toString.call(str);//[object String]
- str.\_\_proto\_\_.constructor === String

## 4、["1", "2", "3"].map(parseInt) 为何返回[1,NaN,NaN]

[解惑 \["1", "2", "3"\].map\(parseInt\) 为何返回\[1,NaN,NaN\]\\_清枫草塘的博客-CSDN博客](#)

parseInt() 函数可解析一个字符串，并返回一个整数。

语法

```
parseInt(string, radix)
```

参数	描述
string	必需。要被解析的字符串。
radix	可选。表示要解析的数字的基数。该值介于 2 ~ 36 之间。  如果省略该参数或其值为 0，则数字将以 10 为基础来解析。如果它以 "0x" 或 "0X" 开头，将以 16 为基数。  如果该参数小于 2 或者大于 36，则 parseInt() 将返回 NaN。

返回值

返回解析后的数字。

[https://blog.csdn.net/weixin\\_40688217](https://blog.csdn.net/weixin_40688217)

5、js:将数组扁平化并去除其中重复数据，最终得到一个升序且不重复的数组/二维数组去重解惑 ["1", "2", "3"].map(parseInt) 为何返回[1,NaN,NaN]\_清枫草塘的博客-CSDN博客

parseInt() 函数可解析一个字符串，并返回一个整数。

语法

```
parseInt(string, radix)
```

参数	描述
string	必需。要被解析的字符串。
radix	可选。表示要解析的数字的基数。该值介于 2 ~ 36 之间。  如果省略该参数或其值为 0，则数字将以 10 为基础来解析。如果它以 "0x" 或 "0X" 开头，将以 16 为基数。  如果该参数小于 2 或者大于 36，则 parseInt() 将返回 NaN。

返回值

返回解析后的数字。

[https://blog.csdn.net/weixin\\_40688217](https://blog.csdn.net/weixin_40688217)

6、Set方法去重

- new一个Set对象，set方法去重，set方法只能去重一维数组
- flat方法扁平化，Infinity规定数组长度的限制
- sort方法正序排列，

- 最后把对象转化成数组：

```
1 var array = [1,25,15,[1,2,15,5],15,25,35,1];
2 var set = new Set(array.flat(Infinity).sort((a,b)=>(a-b)));
3 var arr= [...set];
4
   console.log(arr)// [1, 2, 5, 15, 25, 35]
```

## 7、['1', '2', '3'].map(parseInt) 输出什么?并解释

- 1、map是用用来遍历的对数组的每个元素进行处理；
- 2、map参数是一个函数；
- 3、map的返回值是一个新的数组；

```
> ['1', '2', '3'].map(parseInt)
< ▶ (3) [1, NaN, NaN]
```

1, parseInt( '1' , 0)

如果 radix 是 undefined、0 或未指定的，JavaScript会假定以下情况：

1. 如果输入的 string 以 "~~0x~~" 或 "0X" (一个0，后面是小写或大写的X) 开头，那么radix被假定为16，字符串的其余部分被当做十六进制数去解析。
2. 如果输入的 string 以 "~~0~~" (0) 开头，radix 被假定为 8 (八进制) 或 10 (十进制)。具体选择哪一个radix取决于实现。ECMAScript 5 澄清了应该使用 10 (十进制)，但不是所有的浏览器都支持。因此，**在使用 parseInt 时，一定要指定一个 radix。**
3. 如果输入的 string 以任何其他值开头，radix 是 10 (十进制)。

2, parseInt( '2' , 1)

### 返回值

从给定的字符串中解析出的一个整数。

或者 NaN，当

- radix 小于 2 或大于 36，或
- 第一个非空格字符不能转换为数字。

3, parseInt( '3' , 2)

以下例子均返回 NaN:

```
parseInt("Hello", 8); // 根本就不是数值  
parseInt("546", 2); // 除了"0、1"外, 其它数字都不是有效二进制数字
```

## JavaScript 数组

### 创建方法

空数组 `var Obj = new Array();`  
指定长度数组 `var Obj = new Array( Size );`  
指定元素数组 `var Obj = new Array( 元素1, 元素2, ..., 元素N );`  
`var Obj = [ 元素1, 元素2, 元素3, ..., 元素N ];`  
单维数组  
图例 

元素0	元素1	元素2	元素3	元素4	.....	n
-----	-----	-----	-----	-----	-------	---

  
`var a = new Array([数组序列1], [数组序列2], [数组序列N]);`  
多维数组  
图例 

元素	元素	元素	元素	元素	.....	元素
元素0	元素1	元素2	元素3	元素4	.....	n

### 基本操作

存取数组元素  
单维数组 `数组名[下标索引];`  
多维数组 `数组名[外层数组下标][内层元素下标]`  
特性  
数组长度是弹性的, 可自由伸缩  
数组下标从0开始  
下标类型  
数值  
非数值  
转为字符串  
生成关联数组  
下标将作为对象属性的名字  
数组元素可添加到对象中  
增加数组 `使用"[]"运算符指定一个新下标`  
删除数组 `delete 数组名[下标];`  
遍历数组 `for (var 数组元素变量 in 数组)`

### 数组属性

constructor 引用数组对象的构造函数  
length 返回数组的长度  
prototype 通过增加属性和方法扩展数组定义

### 队列方法 先进先出

添加  
`push()` 在数组末尾添加数组  
`unshift()` 在数组头部添加元素  
`concat()` 合并两个数组  
删除  
`pop()` 删除并返回数值的最后一个元素  
`shift()` 删除并返回数组的第一个元素

栈方法  
后进先出

### 子数组

`splice()`  
删除任意数量的项  
1 要删除的起始下标  
2 要删除的项数  
在指定位置插入指定的项  
1 起始下标  
2 0 (不删除任何项)  
3 要插入的项  
替换任意数量的项  
1 起始下标  
2 要删除的项数  
3 要插入的项  
功能 从已有数组中选取部分元素构成新数组

### ECMAScript 3 方法

`slice()`  
参数  
1 返回项的起始位置  
2 返回项的结束位置  
特性  
如果是负数, 则用数组长度加上该值确定位置  
指示位置实为数组的实际下标  
结束位置的实际下标为结束数值减1  
数组排序  
`reverse()` 颠倒数组中元素的顺序  
功能 对字符串或数字数组进行排序  
默认为按字符串比较  
`sort()`  
特性 按数值大小比较  
需函数支持 (升序)  

```
function compare(value1,value2){  
  if (value1 < value2) {  
    return -1;  
  } else if (value1 > value2){  
    return 1;  
  } else {  
    return 0;  
  }  
}
```

  
数组转换  
`toString()` 转换为字符串并返回  
`toLocaleString()` 转换为本地格式字符串并返回  
`join()` 用指定分隔符分割数组并转换为字符串



## 8、描述typeof和instanceof的区别

**typeof:** 获取变量类型, 返回值是字符串;

**Instanceof:** 用于检查一个对象是否是一个构造函数的实例化对象;

## 9、描述Array.slice和Array.splice的区别

直接看官方文档

### a) slice

-- “读取” 数组指定的元素, 不会对原数组进行修改。

语法:

`arr.slice (start, end)`

`start` 指定选取开始位置 (含), `end` 指定选取结束位置。

### b) splice “操作” 数组指定的元素, 会修改原数组, 返回被删除的元素。

语法:

`arr.splice(index, count, [insert Elements])`

`index`: 操作的起始位置, `count = 0`插入元素, `count > 0`删除元素;

`[insert Elements]` 向数组新插入的元素。

## 面试题:

1

```
1 for (i=0, j=0; j<6, i<10; i++, j++) {  
2     k = i + j;  
3 }  
4 console.log(k); //18
```

- for循环的小括号里面有3个语句: 初始化语句、条件判断语句、条件修改语句;
- 如果条件判断语句返回false, 循环结束;

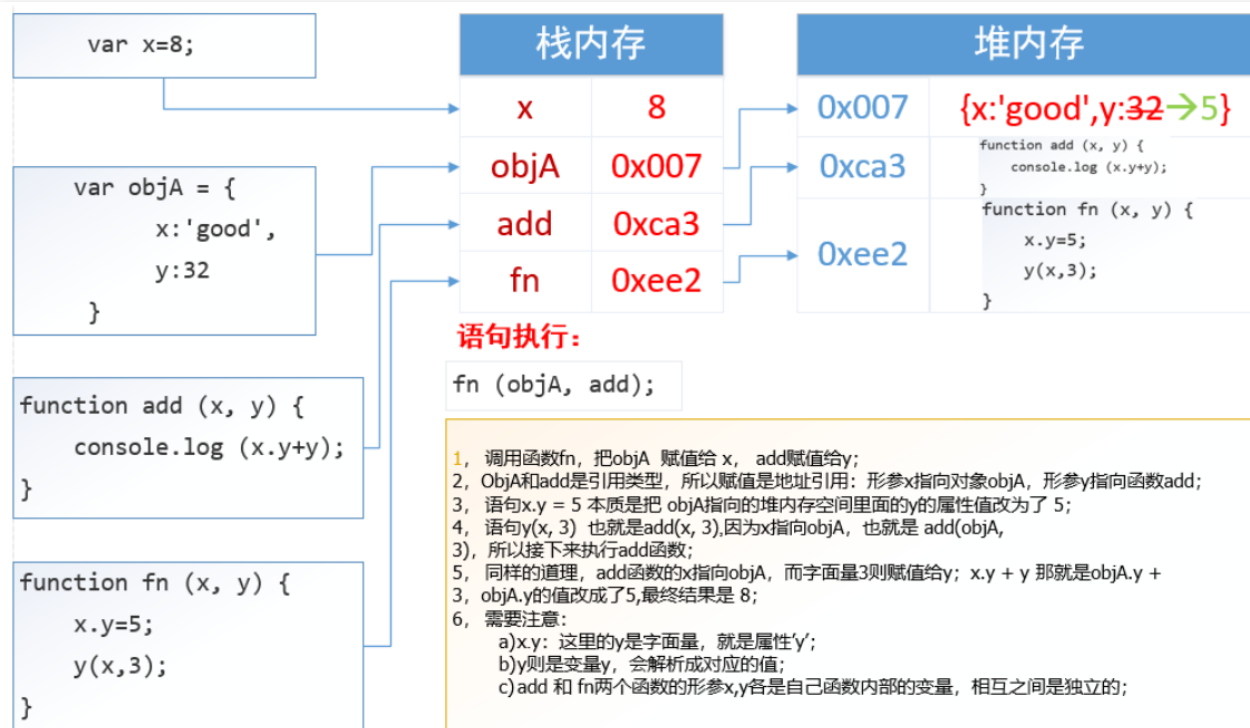
- 逗号(,)运算符返回的是最后一个表达式的值;

```
> for (i=0, j=0; j<12 || i<10; i++, j++) {  
    k = i + j;  
}  
console.log(k);  
22
```

```
> for (i=0, j=0; j<6&& i<10; i++, j++) {  
    k = i + j;  
}  
console.log(k);  
10
```

2

```
1 var x=8;  
2 var objA = {  
3     x:'good',  
4     y:32  
5 }  
6 function add (x, y) {  
7     console.log (x.y+y);  
8 }  
9 function fn (x, y) {  
10     x.y=5;  
11     y(x,3);  
12 }  
13 fn (objA, add);
```



3

```
1 var buttons = [{name:'b1'}, {name:'b2'}, {name:'b3'}];
2     function bind () {
3         for (var i = 0; i < buttons.length; i++) {
4             buttons[i].onclick = function () {
5                 console.log(i);
6             }
7         }
8     };
9     bind ();
10    console.log(buttons);
11    buttons [0]. onclick ();//3
12    buttons [1]. onclick ();//3
13    buttons [2]. onclick ();//3
```

- var声明的变量没有块级作用域;
- 函数内部循环的过程 只是 给每个对象追加一个onclick属性并赋值为函数而已, 一开始是没有执行的;
- 函数名后面跟上小括号表示函数的执行

4

```
1 (function () {
2     var a = b = 3;
```

```
3      })();
4      console.log ("a defined? " + (typeof a !== 'undefined'));
5      //a defined? false
6      console.log ("b defined? " + (typeof b !== 'undefined'));
7      //b defined? true
8      console.log(b);
9      //3
10     console.log (typeof a);
11     //undefined
```

- 函数内部声明的变量是局部变量;
- 如果函数内部直接给一个没有声明的变量赋值, 那么该变量是全局变量;
- !== 表示不完全相等: 'number' !== 'undefined' 返回true